Semantic Vector Space Model: Implementation and Evaluation

Geoffrey Z. Liu
School of Library & Information Science, San Jose State University, One Washington Square, San Jose, CA 95192-0029. E-mail: GLIU@wahoo.sjsu.edu

This article presents the Semantic Vector Space Model (SVSM), a text representation and searching technique based on the combination of Vector Space Model (VSM) with heuristic syntax parsing and distributed representation of semantic case structures. In this model, both documents and queries are represented as semantic matrices. A search mechanism is designed to compute the similarity between two semantic matrices to predict relevancy. A prototype system was built to implement this model by modifying the SMART system and using the Xerox Part-Of-Speech (P-O-S) tagger as the pre-processor of the indexing process. The prototype system was used in an experimental study to evaluate this technique in terms of precision, recall, and effectiveness of relevance ranking. The results of the study showed that if documents and queries were too short (typically less than 2 lines in length), the technique was less effective than VSM. But with longer documents and queries, especially when original documents were used as queries, we found that the system based on our technique had significantly better performance than SMART.

1. Introduction

One of the major problems in the field of information storage and retrieval is how to represent the content of unstructured natural language texts in a more effective way so that better system performance in predicting document-query relevancy can be achieved. Great effort has been made to transcend the limitations of the conventional keyword/inverted file/Boolean logic search paradigm characteristic of the mechanized systems of the 1960s and the early 1970s. Among all the recently-proposed theories, Salton's Vector Space Model (VSM) has proven to be most influential (Raghavan & Wong, 1986; Salton, 1971, 1989; Sparck-Jones, 1971; Sparck-Jones & Tait, 1984). Evidence generated so far indicates that a VSM-based system is at least as good as, and sometimes better than, conventional IR systems in an experimental environment. However, some difficulties with VSM have also been voiced (Cooper, 1991; Sutcliffe 1991; Wong, Ziarko, Raghavan, & Wong, 1987; Wong, Ziarko, & Wong, 1985).

The major difficulty with VSM is its inadequacy in disambiguating the meaning of terms used in natural language texts, which is a direct consequence of the oversimplicity of its purely term-based representation of content. In this model, keywords are identified, taken out of context, and further processed to generate term vectors. All the non-keyword terms, syntactic structures, and other linguistic elements of text are discarded. However, meaning is conveyed not only by keywords, but also by other linguistic elements such as functional words and syntactic patterns. In natural language, rich compositional structures are employed to form low-level contexts to differentiate the lexical meaning of keywords (concepts), to denote the relationships between semantic entities, and finally to convey the semantic content of texts. These additional linguistic elements carry a significant amount of meaning in natural language texts, probably as much as the keywords. An information system should be able to capture some of the information conveyed by these linguistic elements if its performance is to be improved significantly.

The existence of such problems led to the belief that VSM had reached the limit of its effectiveness (Smeaton, 1986, 1989). Some researchers have completely abandoned VSM, hoping to find a new IR approach purely based on the techniques of natural language understanding. These researchers tend to believe that a better understanding of meaning is essential for retrieving relevant information. Especially in a question-answering environment, an IR system has to *understand* both the query and texts in the database. The system should be able to reply to a user's request by actually ''answering'' the query

Received September 29, 1995; revised January 19, 1996; accepted April 22, 1996.

rather than merely trying to identify documents/texts that are likely to be relevant to that query. To achieve the required level of understanding, the system has to include a large amount of semantic processing and sophisticated knowledge representation.

Most of the studies done under this philosophy involve some ad hoc designs of knowledge-representing schemes, enormous investment in knowledge engineering, and complicated meaning representation. Because of the limitation of current semantic processing and knowledge engineering techniques, such systems are not likely to have much practical significance, not at least in the near future (Brajnik, Guida, & Tasso, 1988; Cohen & Kjeldsen, 1987; Dillon & Gray, 1983; Doszkocs, 1986; Evans, Gunther-Webster, Hart, Lefferts, & Monarch, 1991; Fagan, 1987a, 1987b, 1989; Fuhr & Buckley, 1991; Hahn & Reimer, 1986; Lin, 1991; Lewis & Croft, 1990; Lochbaum & Streeter, 1989; Metzler & Hass, 1989; Rau, 1987; Rau, Jacobs, & Zernik, 1989; Sacks-Davis, Wallis, & Wilkinson, 1990; Schwarz, 1990; Sheridan & Smeaton, 1992; Smeaton & Sheridan, 1990; Smeaton & van Rijsbergen, 1988; Smeaton, Voutilainen, & Sheridan, 1990; Sparck-Jones & Kay, 1973; Thiel & Hammwohner, 1987; Turtle & Croft, 1991; Wilkinson & Hingston, 1991; Wood & Sommerville, 1988; Yao & Wong, 1991).

This article presents a text representation and searching technique called ''Semantic Vector Space Model'' (SVSM), which is a combination of Salton's VSM with heuristic syntax parsing and distributed representation of semantic case structures. In this model, both documents and queries are represented as semantic matrices. Retrieval of relevant documents is achieved by computing similarity values between semantic matrices. A prototype system was built to implement and evaluate the model in terms of precision, recall, and effectiveness of relevance ranking. The results of the study indicate that this technique promises a way of abstracting and encoding richer semantic information of natural language texts to improve precision without involving sophisticated semantic processing.

The focus of this article is the implementation and evaluation of the technique. We begin with a brief introduction to the theoretical model, and then give a detailed description of our implementation. A series of experimental studies conducted to evaluate the effectiveness of this technique is reported. With an awareness of potential biases and limitations of the study, conclusions are drawn. Finally, a brief discussion of possible directions for future investigation is given.

2. Review of Related Studies

The possibility of combining VSM with a distributed representation of semantic case structures was investigated in at least two studies (Sutcliffe, 1991; Wendlandt & Driscoll, 1991). In Wendlandt and Driscoll's study, a set of 60 semantic categories consisting of attributes and thematic roles were added to the end of the term vectors for documents and queries. These semantic categories were not related to indexing tokens per se. As in VSM, weights were assigned at the document level to both terms and semantic categories. The similarity of a document to a query was calculated by summing the products of the elements of the document and query vectors term/category-paired.

While Wendlandt and Driscoll (1991) attempted to use semantic categories (attributes and thematic roles) directly as indexing tokens in document-level representation, Sutcliffe (1991) used thematic roles as a means of differentiating the distributed representation of the lexical meaning of concepts. For any given concept represented by an indexing token, a distributed representation was first constructed by assigning a centrality value to each of the pre-defined semantic dimensions (or ''features'' as he called then) to indicate whether, or to what degree, a semantic dimension applied to that concept. Similarly, the case relationships realized by each concept in a specific sentence were represented as a distributed pattern over case categories, a vector in which each element indicates the strength of the concept's link to a specific case category. Then the vector representation of a concept and the vector description of its case relationships were related and combined into a rectangular matrix by computing the outer product. Matrices generated for each sentence that represent individual case-filler pairs were summed to yield a new matrix of the same dimension. This new matrix constitutes a higher level structure, a case frame representation of the sentence in question. Searching for relevant information was achieved by matrix comparison, by computing the dot product of each row in one of the given matrices with its corresponding row in the other, and then summing up the results. The total of row vector dot products was taken as a measure of the similarity between the two matrices and was interpreted as the prediction of relevancy.

Both these two studies suffered from some problems. A thorough discussion is found in Liu (1994). The following summarizes some of the observations.

The major problem is their lack of a mechanism for automatically deriving semantic case realizations of concepts from texts at the sentence *and* document levels. Thematic roles were assigned to concepts (indexing tokens) manually and weighted arbitrarily. In Wendlandt and Driscoll's study (1991), if a case trigger is associated with multiple case categories, each category was assumed to be realized with equal probability at each occurrence of the trigger. In Sutcliffe's research (1991), centrality values were again manually assigned to a set of conceptual features and a concept was assumed to be associated with only one case category, with a strength value arbitrarily set to 5 on a 10-point scale. Without an automatic mechanism for deriving and abstracting semantic case

realizations of concepts, the approach remains useless for the purpose of information retrieval. This is true regardless of how effective it is in representing and differentiating the semantics of texts.

Another critical problem in Wendlandt and Driscoll's study (1991) is that to derive higher-level representation of texts, probability values assigned for semantic case realizations were manipulated with no consideration of to which tokens they were attached. The links between semantic case categories and tokens were completely ignored. Document-level case category weights derived this way represent only the overall frequency pattern of thematic roles, and are hardly useful in representing the semantic content of texts.

In Sutcliffe's study, the researcher only offered a way of representing texts at the sentence level. It was assumed that queries (and the documents as well, though not made explicit) would not be longer than one sentence. In a free text IR environment, natural language queries can easily be longer and documents definitely include many sentences. It was not clear how multiple matrix representations at the sentence level could be converted into a structure at the higher level to represent a body of text consisting of multiple sentences.

Although these studies suffer from these problems, they lead us in a direction worthy of further exploration. It is evident that the concept of semantic case structure has the potential of being used for meaning representation, and that an effective text representing and searching technique can be developed based on this concept.

3. Distributed Representation of Semantic Case Structure

The concept of semantic case structure is based on Filmore's case theory (Filmore, 1966, 1968, 1969a, 1969b, 1970, 1971, 1982). Filmore's original proposal was that the semantic content of a sentence could be captured by specifying the action being described, with further arguments saying more about the action from different perspectives. Such arguments were categorized as a set of semantic functions called ''cases'' by Filmore, and ''thematic roles,'' ''theta roles,'' ''semantic cases,'' and ''semantic roles'' by other linguists (Anderson, 1971; Blake, 1930; Chierchia, 1988; Chomsky, 1981; Dowty, 1989, 1991; Jackendoff, 1972, 1990). A list of the thematic case roles compiled from the literature and implemented in our study is given in Table 1.

Case roles designate the semantic relations between a predicate and other semantic entities involved in the event described in a sentence. Such relations roughly correspond to the relations of concepts and comprise most of the contextual information at the sentence level. By tagging words with their thematic role categories, it is possible to capture and encode contextual information at the sentence level. Therefore, with a small number of semantic case categories, the semantic case structure of a sentence (i.e., the contextual information at the sentence level) can be represented. If such information can be extracted from texts based on syntactic structures, it can be used to enhance information retrieval.

TABLE 1. List of implemented thematic role categories.

Accompaniment	Consequence	Patient
Action	Content	Process
Agent	Destination	Purpose
Alternative	Direction	Quantity
Aspect	Exemplary	Range
Attribute	Experiencer	Recipient
Beneficiary	Extent	Reference
Cause	Instrument	Source
Comparison	Location	State
Component	Manner	Time
Condition	Means	Undefined

The possibility of recovering semantic case structure from the syntactic pattern of a sentence lies in that if the syntactic pattern of a sentence is derived from and determined (though not uniquely) by its underlying case structure, then the relationship between the syntactic pattern and the case structure cannot be arbitrary (Bailin & Grafstain, 1991; Bowerman, 1990). Although complete recovery of the semantic case structure is impossible without fully interpreting the sentence, we can predict it from the syntactic pattern (Carrier-Duncan, 1985; Rosen, 1984; Zaring, 1991; Zernik & Jacobs, 1990). Most linguists recognize the close correspondence between some syntactic constituents and certain thematic roles (Dowty, 1989). A table was constructed by James Allen to summarize such correspondence (Allen, 1988, p. 205).

Syntactic constituents tend to be associated with more than one thematic role. For instance, the thematic roles ''AGENT'' and ''EXPERIENCER'' are often taken by subjects if the sentences are active, or by the noun phrases preceded by the preposition ''by'' if the sentences are passive. The thematic role ''PATIENT'' is often taken by objects if the sentences are active or by subjects if the sentences are passive. Although this syntactic pattern can be exploited to eliminate other possibilities, it is difficult to determine which case actually occurs without semantically interpreting the whole sentence. In an active sentence, the thematic role taken by the subject can be either ''AGENT,'' ''THEME,'' or ''EXPERIENCER.'' In the case of a passive sentence, while the subject is most likely to take the case role ''PATIENT,'' it is not clear which one of the several possible case roles (''AGENT,'' ''MEANS,'' ''INSTRUMENT,'' and ''TIME'') is taken by the noun phrase preceded by the preposition ''by.'' As to the other thematic roles, they are often realized by prepositional phrases in English and relatively stable relationships seem to exist with prepositions as case markers, though the relationships are always many-to-many.

Such difficulties make it impossible to accurately pre-

dict thematic role realization. One reasonable solution is to sacrifice the accuracy of thematic role assignment. Instead of trying to identify *the* thematic role each semantic entity takes from its syntactic function and recover *the* semantic case structure of a sentence from its syntactic pattern, we can try to make a wise guess of what thematic role(s) a syntactic constituent is likely to realize and assign thematic roles in a probabilistic manner based on the information gathered from a simple analysis of the syntactic structure.

Assume that we have a set of registers for every syntactic constituent of a given sentence, each corresponding to one of the pre-defined thematic roles. If it is determined from syntactic clues that a constituent realizes a specific thematic role, the register corresponding to this thematic role is set to one and all the others to zero. However, if it is not certain which thematic role the constituent realizes, the registers can be set to some values indicating the probabilities of its linking to different thematic roles.

How to define these probabilities is the problem. The solution is not as difficult as it appears. For the main verb (predicate) of a sentence, it can be assigned the case role ''ACTION'' if it is not the special ones like ''be'' and ''have.'' For the subject and object, the possibilities can often be reduced to two or three thematic roles and equal probabilities can be assigned to each of them. Finally, for prepositional phrases, the probabilities of their thematic role realizations can be determined by analyzing the probability distribution in general. The relationships between thematic roles and prepositions as thematic role triggers appear to be in certain patterns of probability distribution in the English language (probably in other languages as well). By analyzing the frequencies of preposition-case pairs in a fairly large sample of texts, a reasonable approximation can be calculated and a prepositional case realization (PCR) probability dictionary can be constructed.

The result of representing semantic case realization of syntactic constituents in such a probabilistic way is a distributed representation of semantic case structure at the sentence level. The representation captures low-level contextual information in texts and can be used to augment and differentiate the lexical meaning of indexing tokens and, therefore, help improve system performance. The following section explains how a document-level representation can be derived from such low-level structures and how similarities can be computed based on this kind of representation.

4. The Semantic Vector Space Model

For each sentence in a given text $\boldsymbol{D}$, after syntax parsing is done, a set of values are assigned to each content word indicating the probabilities of its triggering thematic roles at the sentence level. For the ease of exposition, we call such values ''case weights.''

As in VSM, content words are stemmed to generate tokens and non-content words, i.e., grammatically functional words, are dropped from further consideration. An importance value is calculated for each token based on its in-document and collection-wide frequencies as an index of its discriminating power. Tokens with importance values equal to, or higher than, a predefined threshold are used as indexing tokens for the collection. The rest are either given some special treatment or simply discarded (Salton & McGill, 1983).

Since case weights are assigned on a token basis, a word would have as many case weight assignments as the number of times it occurs in the given text. Therefore, regardless of whether a stemming procedure is involved, for each unique token chosen for indexing, there will be more than one case weight assignment in general.

Assume m thematic role categories are considered. For the ith occurrence of an indexing token $\boldsymbol{T}$, the case weight assignment can be described as a vector $\boldsymbol{V}_i$,

$$\boldsymbol{V}_i = (w_{i1}, w_{i2}, w_{i3}, \ldots, w_{im})$$

where w_{ij} is the case weight corresponding to the jth thematic role category.

If the in-document frequency of token $\boldsymbol{T}$ is k, then we have k case weight assignments, and the document-wide case weight assignments of token $\boldsymbol{T}$ can be conveniently expressed as a $k*m$ matrix $\boldsymbol{M}_T$,

$$\boldsymbol{M}_T = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{km} \end{bmatrix}.$$

Each row of the matrix corresponds to one occurrence of the token $\boldsymbol{T}$, and each column corresponds to the distribution of the token's realizations of a thematic role within the text.

Then for token $\boldsymbol{T}$ with regard to document $\boldsymbol{D}$, we define a new vector $\boldsymbol{V}_T$ by seeking a linear combination of the row vectors of the matrix,

$$\boldsymbol{V}_T = (c_1, c_2, \ldots, c_m)$$

where $c_j = (1/k) \sum_{i=1}^{k} w_{ij}$, corresponding to the jth thematic role category, and $j = 1, 2, \ldots, m$. $\boldsymbol{V}_T$ describes the document-level case structure pattern of the indexing token $\boldsymbol{T}$ in the text $\boldsymbol{D}$, and is called a *case vector.*

While the case vector differentiates the meaning of a token by describing its case structure pattern, it says nothing about the significance of the token in discriminating this text from the others in the collection. To deal with this problem, we extend the case vector by adding the weight of the token as defined in VSM. In other words, for a token $\boldsymbol{T}$ with regard to the text $\boldsymbol{D}$, we define a vector $\boldsymbol{V}$ based on its case vector and significance weight,

$$\boldsymbol{V} = (w, c_1, c_2, \ldots, c_m)$$

where w is the token's significance weight, and c_j is still the token's case weight as defined in the case vector, $j = 1, 2, \ldots, m$.

The vector $\boldsymbol{V}$ defined above is intuitively called a *token semantic vector*. It constitutes a distributed representation of the semantic information conveyed by token $\boldsymbol{T}$ at the document level. The first element of a token semantic vector (the significance weight w) describes the importance of the token in defining the content of the particular text and its discriminating power with respect to a specific collection. Its value is determined by the token's frequency distributions in that text *and* within the whole collection. The other elements of the token semantic vector (the case weights c_j, $j = 1, 2, \ldots, m$) describe the pattern of semantic case realizations by the token at the document level. The values of these elements are determined by the frequency distribution of semantic case realizations by the token *within* that particular document, and they are independent of both the other documents in the collection and the collection as a whole.

Note that for each indexing token, the token semantic vector is uniquely defined for a given text. Assume that n tokens are used to index a collection. For a given text $\boldsymbol{D}$, each indexing token will have an unique token semantic vector, and for that text we would have n well-defined token semantic vectors, each of which corresponds to one indexing token.

That is, for any of the n indexing tokens $\boldsymbol{T}_i$, there is a token semantic vector $\boldsymbol{V}_i$,

$$\boldsymbol{V}_i = (w_i, c_{i1}, c_{i2}, \ldots, c_{im})$$

where w_i is the significance weight of the token $\boldsymbol{T}_i$, c_{ij} is the case weight of the token with regard to the jth thematic role category, $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. Combining these n token semantic vectors by cross product defines a $n*(m + 1)$ matrix $\boldsymbol{M}_D$,

$$\boldsymbol{M}_D = \begin{bmatrix} w_1 & c_{11} & c_{12} & \cdots & c_{1m} \\ w_2 & c_{21} & c_{22} & \cdots & c_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_n & c_{n1} & c_{n2} & \cdots & c_{nm} \end{bmatrix}$$

where the first column is actually the vector representation of a document in VSM, and the rows are token semantic vectors for each indexing token defined in the given text. $\boldsymbol{M}_D$ is a matrix representation of the semantics of text $\boldsymbol{D}$, and is called a *semantic matrix*.

Both query and document texts can be represented as semantic matrices. The problem of relevancy prediction becomes a problem of matrix comparison, a problem of calculating the similarity between two semantic matrices. After trying several approaches to solving the problem of matrix comparison without success (Liu, 1994), including the one proposed by Sutcliffe (1991), we came to the following solution.

Instead of attempting direct matrix comparison, the similarity between two semantic matrices is defined as the cosine of the angle formed by vectors derived from such matrices.

Given a semantic matrix $\boldsymbol{M}_T$, a vector is defined by distributing the term weight for each indexing token across its corresponding case vector and taking the products of term weights and case weights as elements. Using the same notation, the vector is defined as

$$\boldsymbol{V}_M = (w_1c_{11}, w_1c_{12}, \ldots, w_1c_{1m}, w_2c_{21}, w_2c_{22}, \ldots, w_2c_{2m}, \cdots w_nc_{n1}, w_nc_{n2}, \ldots, w_nc_{nm}).$$

The vector $\boldsymbol{V}_M$ is called a *text semantic vector*. For each element, $w_i c_{ij}$, w_i is the term weight of the ith indexing token and c_{ij} is the case weight of the token corresponding to the jth thematic role category. Conceptually, the product $w_i c_{ij}$ describes the significance of token $\boldsymbol{T}_i$ realizing the jth thematic role.

Assume we have a query $\boldsymbol{Q}$ represented as a query semantic matrix $\boldsymbol{M}_Q$, and a document $\boldsymbol{D}$ represented as matrix $\boldsymbol{M}_D$,

$$\boldsymbol{M}_Q = \begin{bmatrix} r_1 & q_{11} & q_{12} & \cdots & q_{1m} \\ r_2 & q_{21} & q_{22} & \cdots & q_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_k & q_{k1} & q_{k2} & \cdots & q_{km} \end{bmatrix}$$

$$\boldsymbol{M}_D = \begin{bmatrix} s_1 & d_{11} & d_{12} & \cdots & d_{1m} \\ s_2 & d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_k & d_{k1} & d_{k2} & \cdots & d_{km} \end{bmatrix}.$$

The query text semantic vector and document text semantic vector derived from matrices $\boldsymbol{M}_Q$ and $\boldsymbol{M}_D$ are correspondingly,

$$\boldsymbol{V}_Q = (r_1q_{11}, r_1q_{12}, \ldots, r_1q_{1m}, r_2q_{21}, r_2q_{22}, \ldots, r_2q_{2m}, \cdots r_nq_{n1}, r_nq_{n2}, \ldots, r_nq_{nm})$$

$$\boldsymbol{V}_D = (s_1d_{11}, s_1d_{12}, \ldots, s_1d_{1m}, s_2d_{21}, s_2d_{22}, \ldots, s_2d_{2m}, \cdots s_nd_{n1}, s_nd_{n2}, \ldots, s_nd_{nm}).$$

Let θ be the angle formed by $\boldsymbol{V}_D$ and $\boldsymbol{V}_Q$, $\boldsymbol{s}$ and $\boldsymbol{r}$ be the term vectors (the first columns) of $\boldsymbol{M}_D$ and $\boldsymbol{M}_Q$, $\boldsymbol{d}_{i*}$ and $\boldsymbol{q}_{i*}$ be the case vectors of the ith row in $\boldsymbol{M}_D$ and $\boldsymbol{M}_Q$ respectively, i.e., $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$, $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$, $\boldsymbol{d}_{i*} = (d_{i1}, d_{i2}, \ldots, d_{im})$, and $\boldsymbol{q}_{i*} = (q_{i1}, q_{i2}, \ldots, q_{im})$. The similarity between the two semantic matrices $\boldsymbol{M}_D$ and $\boldsymbol{M}_Q$ is defined in the following theorem.

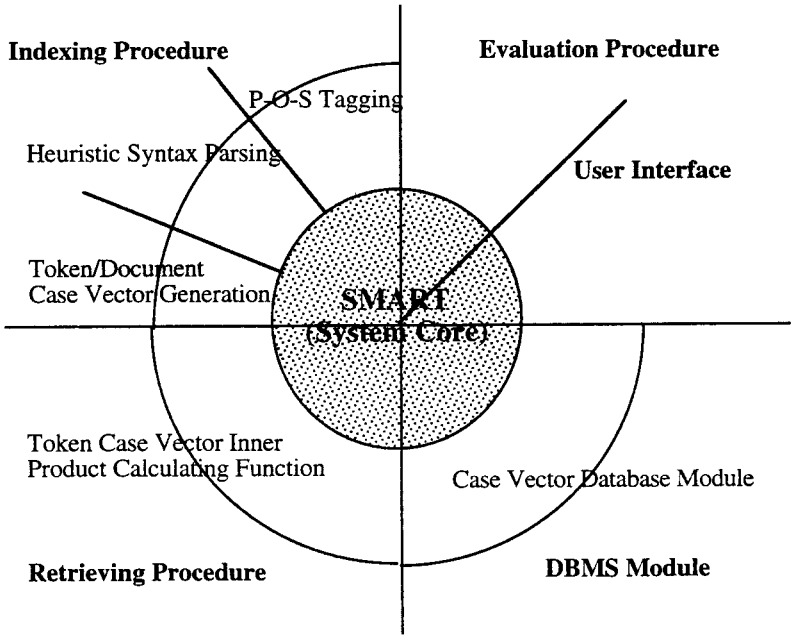

FIG. 1. Implementation of SVSM.

Theorem of Cosine Measure

If the term vectors and case vectors are normalized to unitary length, i.e., $\|\boldsymbol{s}\| = \|\boldsymbol{r}\| = \|\boldsymbol{d}_{i*}\| = \|\boldsymbol{q}_{i*}\| = 1$, then,
i) vectors $\boldsymbol{V}_D$ and $\boldsymbol{V}_Q$ are also normalized to unitary length;

ii) $\cos\theta = \sum_{i=1}^{n} s_i r_i (\boldsymbol{d}_{i*} \cdot \boldsymbol{q}_{i*})$.

The cosine measure defined in this theorem is used to calculate the similarity between two semantic matrices. As in VSM, the similarity value is interpreted as a prediction of relevancy. The proof of this theorem is given in Appendix A.

5. Implementation of SVSM—The SMART++ System

SVSM was implemented in a prototype system called ''SMART++,'' which is basically an extension of the SMART system, Version 10.1 (Salton, 1968, 1989, 1991; Salton & Buckley, 1991; Salton & McGill, 1983). As illustrated in Figure 1, SMART++ consists of five main blocks: The indexing procedure, the retrieving procedure, the database procedure, the user interface, and the evaluation procedure. The evaluation function, database accessing components, user interface, and some low-level pre-processing functions were directly adopted from SMART. Modification was mainly done to the indexing and retrieving functions. Details of the implementation are given below, with emphasis on the added functional components and modification done to SMART.

5.1. The PCR Probability Dictionary

In order to assign case categories to prepositional phrases probabilistically, we need to have information about the general pattern of prepositional case-role realization in the English language. A PCR probability dictionary had to be constructed.

A corpus of English texts was constructed out of 358 articles (about 7,154 sentences) from 12 different news groups. Care was taken to insure that different subject areas were represented and that each news group had enough articles for selection. Articles were not chosen randomly, but rather according to the following rules: 1) The article should be well composed and reasonably similar to the style of published texts; 2) it has to be at least 20 lines long; and 3) it should be written by a native English speaker. Necessary editing was conducted to convert the texts into a standard format so that they could be read by a small system designed to facilitate the encoding process, and also to delete lines quoted from previous postings so that the same passages would not be encoded more than once.

Texts in the corpus were encoded sentence by sentence. At each session, a sentence was displayed, and the analyst was prompted to assign one of the standard case categories (see Table 1) to each occurrence of a preposition in the sentence according to his understanding of the prepositional phrase in relation to the predicate or other seman-

tic constituents. A manual compiled from several English grammar textbooks and dictionaries was used through the encoding process (see Appendix B).

The probability data were generated from the encoding done by the author himself. Although the original plan was to have three analysts doing the encoding, lack of funding made it impossible.

5.2. General Design of the Indexing Process

In SVSM, a text is represented as a semantic matrix. The first column of the matrix is the term vector as defined in VSM, and for each indexing token there is a case vector linked to it. It was natural for us to use the indexing function of SMART to generate term vectors and to make case role analysis a completely separate process. Indexing a collection using the SVSM technique became a two-step task, first to run the indexing procedure of SMART to generate term vectors for all the documents in a collection, and then to execute the case role analysis subsystem on the same collection to generate case vectors for all the identified indexing tokens, document by document. The case role indexing process can be divided into four distinct steps: Part-of-speech tagging, syntax parsing, sentence-level case role assignment and weighting, and finally generation of document-level representations. Each step is a well-defined subtask implemented as an individual function.

5.3. Part-Of-Speech Tagging and Syntax Parsing

Part-Of-Speech (P-O-S) tagging is done independently. Texts and sample queries are P-O-S tagged all at once before they are fed into the indexing procedure, using the Xerox Part-Of-Speech Tagger Version 1.0 (Cutting, Kupiec, Pedersen, & Sibun, 1992; Cutting, Pedersen & Halvorsen, 1991). The tagger was built in ANSI Common Lisp and was based on a hidden Markov model. It was reported by Cutting et al. (1992) that 96% accuracy was achieved when using a lexicon and tag set built from the tagged text of the Brown corpus.

Syntax parsing is handled by a heuristic syntax parser. The reason for not using the well-known CFG and RTN syntax parsing techniques is that a CFG- or RTN-based parser would introduce unnecessary complexity and compromise the system's efficiency by providing superfluous functionalities. For instance, such a syntax parser would construct all possible interpretations of the syntactic structure of a sentence when it cannot be uniquely determined, and at the same time check the grammaticality of the sentence. Both of these functionalities are not necessary in our case.

Issues of phrase attachment. Ambiguity of syntactic interpretation is often caused by uncertainty of phrase attachment. While generating all the possible interpretations of a syntactic structure is unnecessary for implementing SVSM, phrase attachment does constitute an important issue and requires some clarification.

Although the semantic case realization of a phrase is *in relation to* the syntactic constituent to which the phrase is attached, information regarding phrase attachment is not included in the text representation. In SVSM, the link between the syntactic constituent and the phrase of attachment (the modified and the modifier) is not preserved.

One may argue that not preserving the relationship of phrase attachment may discount the effectiveness of SVSM. Our understanding is that, as with any indexing algorithm, the process of indexing is one of information filtering, and therefore, one of information loss. The fundamental issue here is how much information we can afford to lose in order to capture and represent sufficient information for effective retrieval of relevant texts. Whether adding phrase attachment information significantly improves the effectiveness of the model, i.e., whether not preserving the relationship of phrase attachment is too big an information loss, has to be determined with further investigation. However, it seems technically difficult to preserve the relationship of phrase attachment in the document-level representation. This is because phrase attachments are highly syntax-dependent and sentence-specific. Although a different scheme (such as term networks) can be used to represent phrase attachments at the document level, the indexing and retrieval mechanism based on the combination of the scheme and the SVSM representation may become too complicated to be practical.

An additional concern is whether attaching a phrase to different syntactic constituents affects the prediction of semantic case realization by this phrase. It appears that attaching non-prepositional phrases (infinitive and participle phrases) to a different syntactic constituent does not change their semantic case realization in the sentence, and therefore prediction of semantic case realization by such phrases can be achieved without considering the relationship of phrase attachment. However, this is not true for prepositional phrases.

With prepositional phrases, the relationship between semantic case realization and phrase attachment is unpredictable. In one situation, the attachment of a prepositional phrase to different syntactic constituents may change the thematic role of the phrase, while in another situation the thematic role of the prepositional phrase remains the same regardless of the syntactic constituent to which it is attached.

For instance, in the sentence ''She decorated the tree with flowers,'' the prepositional phrase ''with flowers'' can be either attached to the predicate ''decorated'' or to the object ''the tree.'' In the former case, ''flowers'' becomes something used for decoration and the prepositional phrase takes the thematic role ''INSTRUMENT.'' In the latter case, it means that the tree she chose to

decorate has flowers on it and the thematic role of the prepositional phrase becomes ''COMPONENT.'' However, in the sentence ''He called the girl from England,'' the prepositional phrase ''from England'' takes the same thematic role ''SOURCE'' regardless of whether it is attached to the object or to the predicate.

It is tempting to assume that prediction of semantic case realization by prepositional phrases might be improved if information of prepositional phrase attachment is utilized. However, this possibility is undermined by difficulties, mainly due to the lack of theoretical understanding of the relationship between prepositional phrase attachment and semantic case realization. It is not clear if all of the prepositions in English or just some of them are dependent on phrase attachment in terms of semantic case realization. Furthermore, for those that are dependent on phrase attachment, we do not know if there is a stable pattern that can be exploited for better prediction. Finally, utilizing such information for automatic semantic case analysis would require thorough interpretation and disambiguation of phrase attachment, which is impossible to achieve without involving extensive semantic processing.

The heuristic syntax parser. The heuristic syntax parser works in a mixed mode, starting in the top-down mode by dividing a sentence into segments, and then switching to the bottom-up mode to process the segments one by one. The process is divided into three phases: Segmentation, segment parsing, and main structure recognition. Figure 2 shows how a sentence is parsed step by step. Some sample results generated by the parser are given in Appendix D.

During the initial phase, the parser divides a sentence into several segments using prepositions (excluding ''of'), relative pronouns, and some other clause or phrase markers as dividing points, going from left to right. If one dividing point immediately follows another, the second one is ignored. If a dividing point is found at the end of the sentence, the word is either a particle or a preposition modifying the leading relative pronoun, and it is simply attached to the last segment of the sentence. If the sentence begins with a dividing point, the first segment is just like the other segments. Otherwise, all the words from the beginning of the sentence are put into the first segment until a dividing point is found.

Every time a segment is generated, a parameter in the ''Segment'' data object called ''Segment Type'' is set according to the first token of the segment (which is a dividing point) to indicate what syntactic element the segment presumably contains.

The parser switches to the bottom-up mode once it enters the second phase, where the segments are fine parsed one by one in the reverse order—the last segment first and the first segment last. Parsing within a segment is done from left to right. The main task of this phase is (1) to recognize the syntactic constituent contained in each segment, and (2) to identify and collect the tokens of the main structure of the sentence.

Each segment should contain at least one and only one syntactic constituent (phrase) or a simple sentence, except for the first if the sentence does not begin with a dividing point. Using the parameter ''Segment Type'' as a clue, the parser determines what syntactic constituent it needs to construct and calls the appropriate subroutine to do the work. For instance, if a segment is marked as ''PP-Type,'' a prepositional phrase is expected, and the parser attempts to recognize a prepositional phrase by calling the function ''IS_PP.''

Every time a syntactic constituent is recognized, the constituent is taken out of the segment and stored in an output data structure. (The output data structure is a frame in which slots are reserved for all the syntactic elements that could appear in a sentence as well as for the ''garbage'' collected through the parsing process.) Whatever are left in this segment are then concatenated to the end of the immediately preceding segment—the one that was generated right before the current segment and is the next to parse.

Having finished fine-parsing all the segments, the parser proceeds to the final stage and attempts to construct the main structure of the sentence out of the tokens collected from the previous processing. However, all of the tokens in the sentence might have already been absorbed during the second phase and there might be nothing left for further processing. Whether there are any tokens left or if the collected tokens actually *are* of the main structure of the sentence depends on the success of the first two phases.

Presumably if no error occurs up to this point, the assembled string of words would be the main structure of the sentence—generally a simple sentence. But if the parser errs badly, the tokens collected from the second phase would be nothing but garbage. In either case, an attempt is made to construct a simple sentence out of the string of collected tokens. If the parser succeeds, the recognized syntactic pattern is stored in the output frame. If there are still any tokens left, they are passed out as unprocessed tokens. If the parser cannot recognize any syntactic pattern at all, all the tokens are passed out as unprocessed tokens.

Given the heuristic nature of the algorithms, it is very likely that the parser may fail to recognize the syntactic constituent contained in a segment. In this case, the parser drops the current segment and moves on to the next. The unprocessed segment is taken as a partially recognized syntactic constituent and linked into the output frame in the same way as the successfully constructed syntactic constituent.

The parser contributes even in the worst case where it fails to construct any syntactic constituent of the sentence. In this case, all the segments generated from the initial process are treated as partially recognized constituents.

Phase I

Steps	Input String	DivPoint	SegMarker	Segment	SegType
0	From this has developed the basic network structure among libraries in the United States	"from" (IGNORED)	<NIL>	<NIL>	<NIL>
1	From this has developed the basic network structure among libraries in the United States	"among"	"from"	From this has developed the basic network structure	PP
2	among libraries in the United States	"in"	"among"	among libraries	PP
3	in the United States	<NIL>	"in"	in the United States	PP
4	<NIL>				

Phase II

Steps	Segment	SegType	ParsingFunx	SynxElem	Leftout	Status
1	in the United States	PP	IS_PP	in the United States	<NIL>	S
2	among libraries	PP	IS_PP	among libraries	<NIL>	S
3	From this has developed network structure	PP	IS_PP	from this	has developed network structure	S
4	<NIL>	<NIL>	<POP>	<NIL>	<NIL>	S

Phase III

Steps	InputString	ParsingFunx	Output	SynxElem	Status
0	has developed network structure	IS_SS	<NIL>	<NIL>	I
1	has developed network structure	IS_VP	has developed	VP	S
2	network structure	IS_NP	network structure	VPC	S
3	<NIL>	IS_SS	<NIL>	SB	<SKIP>
4	<NIL>	<POP>	<SYNXFRAME>	<NIL>	S

FIG. 2. Process of heuristic syntax parsing.

Although the words of the main structure are scattered in these segments and mistaken as part of a phrase or a clause, a coarser semantic case analysis is still possible based on the partially-recognized syntactic structure.

The parser mainly consists of a control module that coordinates the parsing process, input/output interfaces, a segmenting procedure, and a group of low-level functions that recognize different syntactic constituents. These

low-level functions are, namely, IS_NP for noun phrases, IS_PP for prepositional phrases, IS_INGP for present participle phrases, IS_PASTP for past participle phrases, IS_INFP for infinitive phrases, IS_VP for predicates, IS_SS for simple sentences, IS_CL for clauses, and IS_QS for questions.

IS_NP and IS_VP are the backbone of the syntax parser. They are implemented as finite state machines and vigorous condition checks are done in them. Number-agreement is imposed within a NP, but not between the subject and VP in a simple sentence.

IS_SS recognizes a simple sentence by first scanning for the leading word of its VP, constructing the VP by calling IS_VP, and then cutting the word string into three parts. Words found prior to the VP, if any, are automatically taken as the logical subject of the simple sentence if VP is active, and as the logical object otherwise. No effort was made to check on its qualification or number agreement with the VP. Then it attempts to construct a NP out of the words following the VP by calling IS_NP. If it succeeds, the generated NP is taken as the VP's complement. Otherwise if the main verb is ''be,'' the function tries to recognize an adjective and takes it as the VP's complement.

The functions IS_CL and IS_QS are implemented based on IS_SS. Both of them adjust the word sequence if necessary to make it appear like a normal simple sentence, and then call IS_SS to finish the work. The functions IS_PP, IS_INGP, IS_PASTP, and IS_INFP are implemented based on IS_NP and do the obvious things.

5.4. Case Assignment and Case Vector Generation

Semantic case analysis and case assignment are done for all of the word strings in the output frame, regardless of whether it is a recognized syntactic element or an unparsed segment or simply the ''garbage'' collected from the syntax parsing process. The process consists of three steps: Case assignment and weighting at the syntactic constituent level, generation of primitive case vectors at the sentence level, and finally making a document-level case representation of the text.

Case assignment and weighting. Case assignment and primitive weighting are first done at the syntactic constituent level and then, if necessary, also within syntactic constituents. The former is called ''blanket case assignment'' and the latter ''secondary case assignment.''

Blanket case assignment is done on both fully recognized constituents and partially parsed segments according to their semantic case realization characteristics (see Table 2). A register is maintained to memorize primitive case assignments and weightings, which are determined either by consulting the PCR dictionary if it is a prepositional phrase, or otherwise by distributing equal probabilities to all the case roles possibly being realized by the constituent. Once the blanket case assignment is done, a slave procedure is called to do the fine processing, to conduct secondary case assignments if necessary, and to convert the result of semantic case analysis into primitive case vectors.

TABLE 2. Syntactic constituent case assignment rules.

Syntactic constituent	Case realization
(Logical) subject	Agent, experiencer
Predicate verb	Action, process
(Logical) object	Patient, recipient
Complement of ''BE''	Attribute
Present participle*	Process, state
Past participle*	State
Infinitive Phrase*	Purpose
Prepositional Phrase	(by PCR dictionary)
Unparsed words*	(by work category)

* Secondary case assignments are done on these syntactic elements.

Secondary case assignment is done on unprocessed word strings, unparsed segments that are not prepositional, and participle and infinitive phrases. Case categories are assigned based on the part of speech of each word. The following rules are used: 1) If the word is a noun, assign the case ''PATIENT'' and ''RECIPIENT'' with equal weights; 2) if it is an adjective, assign the case ''ATTRIBUTE''; 3) if it is a verb, assign the case ''ACTION''; and finally, 4) if the word is an adverb, assign the case category ''MANNER.''

For words without blanket case assignments, secondary case assignments are directly used as terminal results. For words that already have blanket case assignments, secondary case assignments are added to the blanket case assignments.

Generation of primitive case vectors. To make raw case vectors for the content words of a syntactic constituent, the system walks through the word string, taking one word at a time, and ignoring all the words that are either found on the stopword list or cannot be located in the token-concept dictionary constructed by the indexing procedure of SMART.

Words are stemmed before being searched in the token-concept dictionary. If a word is found there, the system proceeds to generate a primitive case vector for this indexing token using the information stored in the case assignment register. The generated raw case vector for each indexing token, consisting of some ''concept_id/case_id/case_weight'' tuples, is written into an internal buffer. Only the non-zero elements of the vector are maintained.

The procedure for generating primitive case vectors gets called once for every syntactic element. All of the raw case vectors generated from the same text are accumulated and maintained in the internal buffer until the end of the text is reached.

Document-level case representation. Having reached the end of a document, the system activates the final module of the indexing procedure to generate document-level case vectors.

With all the primitive case vectors stored in the buffer as ''concept_id/case_id/weight'' tuples, the system sorts the tuples, first by ''concept_id'' and then by ''case_id.'' Then it divides the sum of weights for the same case category of the same indexing token by the in-document frequency of the token. Tuples with the same ''case_id'' and the same ''concept_id'' are combined into one tuple. The result is then written back into the same buffer, starting from the beginning, by overwriting an already-processed tuple. After the final batch of tuples are processed and the result is stored back into the buffer, the rest of the space is cleared.

The internal buffer now contains the non-zero elements of unnormalized document-level case vectors for all the indexing tokens of the text. Once the case vectors are normalized, the document-level case representation of the text is finalized, and the data are written out into the case vector database for the collection.

5.5. Organization of the Case Vector Database

The case vector database consists of a main data file, a primary index, and a secondary index. The main data file is a sequential list of ''case_id/weight'' tuples, first grouped by document ID, and then by concept ID, in ascending order. Tuples within a concept group are sorted by case ID. The primary index is a sorted sequential list. It gives the number of records and an offset value to access the secondary index using document ID as the key. The secondary index is also a sequential list. Each entry contains a concept ID number, an offset value, and the number of records to be read from the main data file. Like the main data file, records in the secondary index are grouped by document ID. Records within each document group are sorted by concept ID.

Both the primary and secondary indexes are kept in memory at run time. To retrieve a case vector from the database, the document ID is used to offset into the primary index, which yields an offset value into the secondary index and the number of records to be examined there. By conducting a binary search, the system can quickly determine whether there is a case vector for the concept, and if there is, where to find it in the main data file.

5.6. Implementation of Retrieving Function

Similarity calculation in SVSM was defined in the cosine measure theorem. By definition, if both term vectors and case vectors are normalized to the unitary length, the similarity of two semantic matrices is the sum of the products of query and document term weights modified by the inner products of their corresponding case vectors. For each indexing token, there is a query term weight and a document term weight. There is also a query case vector and a document case vector associated with each indexing token.

In SMART, term vector similarities, i.e., the inner products of document and query vectors, are computed in the following way. Given two vectors stored in an internal buffer, an inner product computing procedure walks through the vectors, pairs up the weights of indexing tokens, and calculates the product of weights for each indexing token. The product is added to the value previously stored in an accumulating variable, and the result is stored back in there again. By the time it reaches the end of one of the vectors, what remains in the accumulating variable is the inner product of the two term vectors.

In the implementation of SVSM, a new variable is inserted into the code that computes the product of the term weights of an indexing token. What used to be the product of two term weights for the same indexing token is now the product of the term weights *and* the value of the inserted variable. The value of the variable is the value returned by a small procedure, which is either ''1'' or the inner product of the query and document case vectors associated with the indexing token.

The small procedure has two modes. Which mode it assumes depends on whether a switch is turned ''ON'' or ''OFF.'' If it is ''OFF,'' the procedure does nothing but return ''1'' as output, and the system conducts a VSM search. Otherwise, the procedure uses the received document/query ID numbers and the concept ID number to retrieve case vectors from the database and calculates the inner product in exactly the same way as in SMART.

It is necessary to point out that if no case vector is found in the database for a given concept, either the document case vector or the query case vector or both, an error occurs. In this case, the procedure generates an ''error'' message and returns ''1'' as output, assuming optimistically that the two case vectors are perfectly identical.

Theoretically this should not happen. If a concept is used to index a collection by SMART, the concept should have been included in the token-concept dictionary. A case vector should have been generated for the concept if it appears in the text. The assumption here is that the text has not been altered between the two phases of the indexing process and that SMART and SMART++ preparse the text in exactly the same way. However, the second part of the assumption is not guaranteed to be true. This is mainly because of the differences in word parsing between SMART and the Xerox P-O-S tagger. For instance, if two words are linked by a hyphen, SMART takes out the hyphen and treats the two words as one single term, but the P-O-S tagger treats the two words as individual terms.

No effort was made to solve this problem since the tagger is a copyrighted system and there is literally no documentation on its implementation. Fortunately this

TABLE 3. The Cornell experimental collections.

Collection	No. of docs.	No. of queries	Rel. judgment	Subj. field	Source
Adi	82	35	Binary	CIS, LIS	Unknown
Cisi	1,460	112	Binary	CIS, LIS	Unknown
Cran	1,400	225	Binary	Aerodyn.	Cranfield study

problem only occurs in a few special situations and it seems unlikely to be statistically significant.

6. Experimentation and Evaluation

Preliminary experiments were conducted on the prototype system to evaluate SVSM in terms of its relevance prediction power. Using SMART as a benchmark, the evaluation was conducted in two parts, first in terms of precision and recall, and then in terms of the system's relevance-ranking effectiveness.

6.1. Experiments with Cornell Collections

Three Cornell experimental collections (''adi,'' ''cran,'' and ''cisi'') were used in the experiments. The general information about the collections is summarized in Table 3. The results are presented in Tables 4 and 5.

It was found that with the original query samples of the experimental collections, the performance of SMART++ was consistently inferior to that of SMART in terms of precision and recall. To better understand the results, and to insure that the averaged precision and recall values accurately describe the systems' performance, the collections were carefully examined. As the result, some interesting peculiarities came to our attention.

The documents of the three collections are abstracts of research papers, about 12 to 16 lines long. The sample queries for the ''adi'' and ''cran'' collections are extremely short, mostly one-sentence statements stretching up to two lines at most.

The query sample for the ''cisi'' collection consists of three distinct groups. The first 35 queries are exactly the same ones used for the ''adi'' collection and are extremely short. Queries #36 to #57 are of intermediate length, about five to 10 lines long, and are either informative questions or confirmative questions or both. The last 55 queries (#58 to #112) are actual abstracts of research papers that are about 10 to 15 lines long.

All the queries in the first group (#1 to #35) have relevant documents. Among the 21 queries in the second group, 15 queries have relevant documents. In the third group, only 26 out of 54 queries have relevant documents in the collection. Only the queries with relevant documents are included in the calculation of average precision and recall.

Query #101 has only one document (#1204) listed as relevant in the relevance judgment file. After carefully examining both the query and the document, a professor of information science determined that the document was only remotely related to the query and was only marginally relevant. Query #101 is considered to be an outlier.

A survey of the ranked outputs of the retrieval runs gave mixed results. With the ''adi'' and ''cran'' collections whose sample queries are uniformly short, the difference in average precision and recall described the discrepancy in the systems' performance reasonably well. But for the ''cisi'' collection whose sample queries are of different types and lengths, the averaged figures were a poor indicator of actual system performance.

We started by dividing the ''cisi'' sample queries into three groups: #1 to #35, #36 to #57, and #58 to #112. The identified outlier (Query #101) was excluded. All the queries that have no relevant document in the collection were also ignored. For each group of queries, precision and recall were manually calculated at the four top-ranking levels, 5, 10, 15, and 30, and averaged respectively.

No unusual situation was found in the first group. The actual numbers of relevant documents retrieved for the queries in this group did show that SMART++ performed worse than SMART in most cases, as suggested by the differences in average precision and recall. This

TABLE 4. Average recall with original query samples.

No. of docs top-ranked	Collection ''adi''		Collection ''cisi''		Collection ''cran''	
	SMART	SMART++	SMART	SMART++	SMART	SMART++
5	0.4002	0.3106	0.0830	0.0583	0.1903	0.1519
10	0.5166	0.3940	0.1262	0.1052	0.2775	0.2172
15	0.6138	0.5066	0.1539	0.1341	0.3395	0.2807
30	0.7750	0.6840	0.2331	0.1922	0.4498	0.3827

TABLE 5. Average precision with original query samples.

No. of docs considered	Collection ''adi''		Collection ''cisi''		Collection ''cran''	
	SMART	SMART++	SMART	SMART++	SMART	SMART++
5	0.3086	0.2400	0.3447	0.3132	0.2596	0.2036
10	0.2229	0.1686	0.3066	0.2737	0.1933	0.1520
15	0.1752	0.1505	0.2702	0.2395	0.1591	0.1310
30	0.1162	0.1105	0.2197	0.1838	0.1102	0.0919
Average*	0.4637	0.3494	0.1890	0.1514	0.2414	0.1853

* Precision averaged at three recall levels: 0.2, 0.5, 0.8.

agrees with the results for the ''adi'' and ''cran'' collections.

The results for the other two groups are different (Tables 6, 7). The figures for the third group show some slight improvement. It seems to suggest that SMART++ might perform better than SMART with relatively longer queries. However, the results for the second group do not give any support to this expectation, in spite of the fact that the query lengths of these two groups are about in the same range.

This difference in the system's performance might be due to the fact that the sample queries in these two groups are of different types. While it is reasonable to expect keywords and distributed representation of case structures to capture a significant amount of information from the texts of non-question type statements, it might not be necessarily true with question-type queries, especially when the queries are informative questions. If the query is a confirmative question, the keywords and distributed representation of case structures might help since everything is stated explicitly. But if the query is an informative question, it would be difficult for the representation to capture anything useful for relevancy prediction. The query is more focused on the relative pronoun, but the pronoun is treated as a stopword. There is no keyword to link a potentially relevant document directly to the query, and there is no case structure representation for this unknown semantic entity. In this case, both VSM and SVSM have to rely on indirect semantic links provided by the other words in the query. It is very likely that in such a situation, the information captured in the distributed representation of case structures might work to our disadvantage. However, we have no independent data to support this argument.

On the other hand, simple comparison of average precision and recall values could be misleading. This is not only because of possible skewed distributions of precision and recall values, but also because of the potential distortion in the measurement of recall.

To clarify the situation, we calculated the percentage of sample queries with which the system SMART++ performed either ''better,'' ''equally well,'' or ''worse'' than the SMART system. The data are presented in Table 8.

Table 8 shows that the performance of SMART++ was consistently inferior to SMART with the second group of ''cisi'' sample queries. As to the third group, SMART++ was slightly inferior to SMART at the five-document level, but the trend was reversed at all the other three top-rank levels. The data apparently showed bigger differences than what appeared in Tables 6 and 7.

The analysis seems to suggest that with longer queries, especially when the queries are original documents, SMART++ may have better performance. This observation is made based on a post-facto analysis and needs to be verified by more systematic experimentation.

6.2. Measuring Effectiveness of System Relevance Ranking

Although precision and recall are widely accepted measures of system performance, they are insufficient for

TABLE 6. Breakdown analysis of ''cisi'' queries—recall.

No. of docs considered	Queries no. 36–57 (15*)		Queries no. 58–112 (25*)	
	SMART	SMART++	SMART	SMART++
5	0.0667	0.0514	0.1270	0.1031
10	0.1225	0.0980	0.1787	0.1709
15	0.1490	0.1280	0.2156	0.2212
30	0.2290	0.2115	0.2803	0.2971

* Actual number of queries having at least one relevant document in collection.

TABLE 7. Breakdown analysis of ''cisi'' queries—precision.

No. of docs considered	Queries no. 36–57 (15*)		Queries no. 58–112 (25*)	
	SMART	SMART++	SMART	SMART++
5	0.3733	0.2933	0.4077	0.3840
10	0.3533	0.2800	0.2923	0.3120
15	0.2800	0.2489	0.2487	0.2667
30	0.2175	0.1780	0.1853	0.2066

* Actual number of queries having at least one relevant document in collection.

TABLE 8. Breakdown analysis of ''cisi'' queries—percentages.

No. of docs considered	Queries no. 37–57 (15*)			Queries no. 58–112 (25*)		
	Better	Equal	Worse	Better	Equal	Worse
5	2/15	8/15	5/15	3/25	16/25	6/25
10	3/15	5/15	7/15	10/25	9/25	6/25
15	3/15	5/15	7/15	10/25	8/25	7/25
30	2/15	5/15	8/15	12/25	7/25	6/25

* Actual number of queries having at least one relevant document in collection.

evaluating IR systems that predict document relevancy by ranking. The problem is that with precision and recall, we are measuring the number of retrieved relevant documents in reference to the total of relevant documents in a collection and to a predefined top ranking level. The actual ranking within the output set is not considered. But the whole point of ranking is to predict if a document is relevant or more relevant to the given query by putting it higher on the output list than the non-relevant or less relevant ones. In other words, if a document is more relevant than the others, we expect the system to rank it on top of the others. Since precision and recall only account for the number of retrieved relevant documents, not the order of ranking, the measures are inadequate.

To evaluate the effectiveness of a system's relevance ranking, another measure complementary to the concepts of ''precision'' and ''recall'' is needed. The measure used in this study, called ''Ranking Effectiveness Ratio,'' is defined as follows.

Assume that for a given query, we have ranked relevance judgments for all of the relevant documents in the collection. The ranking is in ascending order, with the most relevant document assigned rank ''1.'' The rank of a relevant document on the relevance judgment list is called the document's ''base rank.'' For simplicity, it is assumed that base ranks are unique, i.e., no rank number is assigned to more than one document.

Similarly, assume that for the same query, we have a list of retrieved documents that include all the ones relevant to the query, in which the order of sequence corresponds to the decreasing degree of predicted relevancy. After the non-relevant documents are deleted from the list, only the relevant ones will remain and the sequence of the system's ranking will be well preserved. Rank numbers are assigned to all the remaining documents from top to bottom according to the sequence of the system's ranking in ascending order, starting with ''1.'' The rank of a relevant document assigned on the system's output list is called the document's ''system rank.'' It is assumed that system ranks are unique as well.

For a given number m of relevant documents, let R_{Bi} and R_{Si} be the base rank and system rank for document D_i respectively, $i = 1, 2, \ldots, m$. The ranking effectiveness ratio μ is defined as

$$\mu = \frac{\sum_{i=1}^{m} R_{Bi} \cdot R_{Si}}{\sum_{i=1}^{m} R_{Bi}^2} .$$

The ranking effectiveness ratio μ has the following properties:

i) Generally $\frac{1}{2} < \mu \leq 1$, where $\mu = 1$ when the system ranks of the m documents perfectly match their base ranks. For the worst case where none of the $\frac{1}{2}m(m-1)$ pairs of documents is ranked correctly (i.e., in the same sequence as in the relevance judgment list), $\mu \rightarrow \frac{1}{2}$ when $m \rightarrow +\infty$.
ii) The ratio μ is a monotonic function of the difference between the system rank and the base rank for any document.

The proof of these properties is given in Appendix C. The first property gives the upper and lower bounds of the ranking effectiveness ratio. If one agrees with our interpretation of ranking effectiveness as the total number of document pairs that are ranked in the correct sequence by the system, the second property assures that the measure accurately describes a system's performance of relevance ranking and therefore gives an indexing value of the system's ranking effectiveness.

6.3. Ranked Relevance Judgment Base

To test a system's relevance ranking performance, ranked relevance judgments for all the sample queries are needed. Unfortunately the experimental collections from Cornell University had only binary relevance judgments and we were unable to find ranked relevance judgments for any of these or other experimental collections.

A decision was made to construct a ranked relevance judgment file for the ''cisi'' collection based on the sample of document-type queries and the associated binary relevance judgments. Twelve out of the 25 document-type queries were chosen for this purpose, not randomly, but based on the criteria that the query must have at least five, but no more than 15, relevant documents as listed in the original relevance judgment file. Out of the 12 queries we chose, two have six relevant documents and the other 10 have more than 10 relevant documents. Rele-

vant documents for each query were identified, printed on separate sheets, and stapled together, in random order, with the query sheet as front cover.

Three experts were asked to do the relevance ranking. Each expert was given a complete set of the 12 queries and was asked to rank the documents using the numbering system described earlier. Upon completion, each expert was asked to comment on this mini-project. All the experts complained about the unexpected difficulty of the task. They found that while it was relatively easy to identify the most relevant document and rank the first three or four most relevant, it was difficult to make a judgment on the rest of the documents in terms of degree of relevancy.

One query (Query #82) was found to be extremely unusual. Two experts found that none of the 10 documents listed as relevant in the original relevance judgment was relevant at all. In spite of that, one of the experts still assigned ranks to them, but with a line of written comments pointing out the peculiarity. Another query (Query #96) was found to have a similar problem. An expert judged relevant only one of the several included documents, but decided to rank all the documents anyway.

An additional complication is that one expert assigned equal rank numbers to more than one document at several occasions when these documents were considered to be equally relevant to the given query, while the other two experts consistently assigned unique but consecutive rank numbers to the documents.

In spite of the complications, a ranked relevance judgment base was constructed from the experts' relevance rankings. Unique and consecutive rank numbers were assigned to the documents within each query group. Because of the aforementioned complications, one query group had only two rank assignments for each document. For the other 11 query groups, all the documents had three rank assignments. In any case, a final rank number for each document was derived from the experts' rank assignments.

The documents were first arranged by the average value of their rank numbers in ascending order. If none of the documents had equal average value, this arrangement would generate a well-defined sequence and final rank numbers were assigned to these documents accordingly. However, if two or more documents had the same average value, the following general procedure was used to resolve the ambiguity.

First, the rank assignments of these documents were written as ordered triples (A_i, B_i, C_i), where $A_i \leq B_i \leq C_i$, $i = 1, 2, 3, \cdots m$; m is the total number of documents that had the same average value. If a document only had two rank assignments $a \leq b$, the ordered pair (a, b) was expanded into a triple (a, b, b) by repeating the larger element. Then these documents were arranged into a sequence by sorting the corresponding ordered triples, first by A_i, then by B_i, and then by C_i, in ascending order. If two triples were exactly the same, the order was determined arbitrarily.

The final product was a ranked relevance judgment base for the 12 queries we had chosen for the final stage of our experimentation. The relevance judgment base contains a ranked list of relevant documents for each query. Each document has a unique rank number and the ascending rank number indicates the decreasing degree of document relevancy within the query group.

6.4. Comparison of Relevance-Ranking Effectiveness

The relevance ranking effectiveness of SMART++ was compared with SMART based on the 12 chosen queries. Of the 12 queries, each had two sets of system ranks for its relevant documents—one from the output list by SMART, the other from the output list by SMART++. In either case, the system ranks were paired with the corresponding base ranks from the ranked relevance judgment base and used to calculate the ranking effectiveness ratio according to the formula given earlier.

Consequently, there were 12 pairs of relevance ranking effectiveness ratios. If the data are listed in tabular form, one column would be the measurements of relevance ranking effectiveness for SMART, and the other column for SMART++. A two-tailed Wilcoxon's matched-pair signed-ranks test on the data showed no significant difference between SMART and SMART++ ($Z = -1.2448$, $N = 12$, $p = 0.2132 > 0.05$).

The data used for this analysis were generated by considering all the documents listed in the original relevance judgment file of the ''cisi'' collection. As we mentioned earlier, most of the queries chosen for the ranking effectiveness test have more than 10 documents considered to be relevant by the original relevance judgments. The three experts who did the relevance ranking found that while it was relatively easy to identify and rank the first three or four most relevant documents, it was very difficult to make a judgment on the degree of relevancy for the less relevant ones.

It seems that the rankings of the less relevant documents might not necessarily reflect the actual difference in the degree of relevancy as precisely as with the most relevant ones at the top of the ranking. In other words, the base ranks of the less relevant documents derived from the ranks assigned by the experts might not be as meaningful as the base ranks of the more relevant ones.

Similar and even stronger arguments can be made about the validity of including the queries #82 and #96 in the test. Since these queries were determined by at least one expert to have either none or only one really relevant document out of all those listed in the original relevance judgment file, the relevancy of the other documents is questionable. These documents may not be relevant at all, or at most only marginally related to these queries. This led to the suspicion that using such a ranked

relevance judgment base to evaluate a system's performance might be highly questionable, if not meaningless.

To guard against this possibility, another test was conducted. This time, queries #82 and #96 were excluded from the sample. For each of the remaining 10 queries, only the top five documents listed in the ranked relevance judgment base were considered. Since the relevance rankings of the documents ranked below these five might not be meaningful due to the reasons explained earlier, they were deleted from the output lists in the same way we eliminated the non-relevant documents. Again, rank numbers were assigned to these documents according to their sequence on the output lists. Using these numbers as system ranks, ranking effectiveness ratios were calculated for each query.

The data were once again analyzed statistically using the Wilcoxon test. The result showed that the relevance ranking effectiveness ratios of SMART++ were significantly higher than SMART ($Z = -2.0386$, $N = 10$, $p = 0.0415 < 0.05$). The mean relevance ranking effectiveness ratio for SMART was 0.8309092, and the mean relevance ranking effectiveness ratio for SMART++ was 0.8836363. The system's relevance ranking performance was improved by 6.3%.

6.5. Comparison of SVSM with Generalized Vector Space Model (GVSM)

One of the referees of this article suggested that it might be interesting to compare SVSM with GVSM proposed by Wong and his associates (Wong et al., 1985, 1987). GVSM was presented as a general solution to the term orthogonality problem of VSM. Four test collections (''adi,'' ''crn4,'' ''med,'' and ''medlars'') were used to evaluate GVSM in comparison with SMART and dramatic improvements in precision were reported. The following paragraphs present a brief comparison of these two approaches from the theoretical and empirical perspectives.

Both SVSM and GVSM attempt to extend VSM by incorporating some contextual information in relevance prediction. While SVSM analyzes semantic case structure at the sentence level and encodes this information into its document/query representation, GVSM takes advantage of term co-occurrence patterns that is already captured in the document-term frequency matrix of VSM and builds the information into similarity computation. The kinds of contextual information abstracted and exploited by these two approaches are different—based on different linguistic features and at different semantic levels, although they are utilized for the same purpose of disambiguating indexing concepts.

In GVSM, the orthogonality problem was solved by accepting term dependency and deriving orthonormal basis vectors from the document-term frequency matrix of VSM. Since the similarity measure was based on the derived orthonormal basis vectors, term orthogonality was not assumed, and therefore it does not constitute a theoretical problem. However, in SVSM, the orthogonality of term/case category pairs was implicitly assumed in our design of the matrix similarity measure. How to solve the orthogonality problem of the model and what theoretical constraints this assumption imposes is not clear.

Before comparing the system performance of SVSM with GVSM, we need to point out that the collections used for evaluating GVSM and SVSM are different. The only collection that was used in both studies is ''adi.'' While GVSM was reported to perform consistently better than VSM at every recall level on ''adi'' and all the other collections used in the Wong et al. study (1987), SVSM showed significant improvement only on the ''cisi'' collection with a subset of its sample queries. It was found that with the longer, non-question type sample queries of the ''cisi'' collection, SVSM had better precision and recall than VSM and was significantly more effective in relevance ranking. Relevance ranking effectiveness was not tested in the Wong et al. (1987) evaluation of GVSM.

7. Conclusion

One of the current trends in this field is to apply AI techniques, especially the techniques of natural language processing and understanding, to the problem of information retrieval. With the limitations of semantic processing and the difficulties in content abstraction and knowledge representation, it seems unlikely that the problem of information retrieval can be solved once and for all by completely relying on semantic processing and knowledge representation, and by attempting direct retrieval of information from knowledge bases constructed from natural language texts. One feasible approach is to use the well-developed IR techniques as the backbone and incorporate some NLP techniques to increase the power of content representation without involving sophisticated semantic processing.

In this article, we presented the Semantic Vector Space Model, a text representation and searching technique based on the combination of VSM, heuristic syntax parsing, and distributed representation of semantic case structures. Our study showed that if documents and queries were too short, SVSM was less effective than VSM. But with longer documents and queries, especially when original documents were used as queries, SVSM had noticeably better performance. This suggests that a significant improvement of system performance may be achieved by combining semantic case structure information with the weighted term representation of texts when longer queries are available.

It is understood that the study has limitations. One potential problem is that the PCR probability data were generated from encodings done by the author himself. The English corpus used for this purpose was not con-

structed from randomly selected texts of formal publications, and the size was probably too small as well. P-O-S tagging was done by a separate process, which was awkward and vulnerable to human error. Discrepancies were found between the P-O-S tagger and SMART in word parsing. Some indexing tokens were lost during the process of case indexing, and consequently no case vectors were generated. Although such errors occurred infrequently, it might have contaminated the data, and therefore, damaged the validity of results.

The findings of this study are preliminary and need to be validated by conducting more systematically designed experiments, using different test collections and larger query samples. Nevertheless, it points to several directions for future investigation.

The observation that SVSM works better with longer queries, especially with original relevant documents as queries, is promising. With SVSM, users can use extensive natural language statements to articulate their information needs and ''converse'' with the system more subtly by deliberately using certain sentence structures in their query statements.

The model can be easily adapted for indexing and searching paragraphs and even sentences in a full-text database. Semantic matrices can be constructed at different levels and the same searching mechanism can be employed to retrieve either full texts of documents or related paragraphs or sentences. Relevance feedback and automatic query generation can be achieved in SVSM by adjusting either the weights of a term vector, or the weights of one or more case vectors, or both at the same time. It is also possible to investigate the ideas of artificial neural networks and genetic algorithms in the SVSM framework. It would be interesting to see how the system performs when permutation is done on different parts or in different combinations of elements in a semantic matrix.

Acknowledgments

I thank Drs. Larry N. Osborne and David Chin at the University of Hawaii at Manoa, and Dr. Gerald Lundeen at the University of Tennessee at Knoxville for their generous support of this research. I am especially grateful to Dr. Chris Buckley at Cornell University for sharing his knowledge, and offering precious advice regarding the SMART system and the Cornell experimental collection.

Appendix A: Proof of Cosine Measure Theorem

Theorem of Cosine Measure

If the term vectors and case vectors are normalized to unitary length, i.e., $\|\boldsymbol{s}\| = \|\boldsymbol{r}\| = \|\boldsymbol{d}_{i*}\| = \|\boldsymbol{q}_{i*}\| = 1$, then,

i) vectors $\boldsymbol{V}_D$ and $\boldsymbol{V}_Q$ are also normalized to unitary length;

ii) $\cos\theta = \sum_{i=1}^{n} s_i r_i (\boldsymbol{d}_i \cdot \boldsymbol{q}_{i*})$.

Proof

i) Consider the length of vector $\boldsymbol{V}_D$. By definition we have

$$\|\boldsymbol{V}_D\| = \left(\sum_{j=1}^{m} (s_1 d_{1j})^2 + \sum_{j=1}^{m} (s_2 d_{2j})^2 + \cdots + \sum_{j=1}^{m} (s_n d_{nj})^2 \right)^{1/2}.$$

Note that the expression inside the big parenthesis can be rewritten as

$$\sum_{i=1}^{n} \sum_{j=1}^{m} (s_i d_{ij})^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} s_i^2 d_{ij}^2 = \sum_{i=1}^{n} s_i^2 \sum_{j=1}^{m} d_{ij}^2.$$

Since both the term vector $\boldsymbol{s}$ and case vectors $\boldsymbol{d}_{i*}$ are normalized to unitary length, we have $\sum_{i=1}^{n} s_i^2 = 1$ and $\sum_{j=1}^{m} d_{ij}^2 = 1$ for any $i \in \{1, 2, \ldots, n\}$. Then the expression becomes

$$\sum_{i=1}^{n} \sum_{j=1}^{m} (s_i d_{ij})^2 = \sum_{i=1}^{n} s_i^2 \sum_{j=1}^{m} d_{ij}^2 = \sum_{i=1}^{n} s_i^2 = 1$$

which means $\|\boldsymbol{V}_D\| = 1$.

Similarly, for vector $\boldsymbol{V}_Q$ we have

$$\sum_{i=1}^{n} \sum_{j=1}^{m} (r_i q_{ij})^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} r_i^2 q_{ij}^2 = \sum_{i=1}^{n} r_i^2 \sum_{j=1}^{m} q_{ij}^2 = 1$$

and $\|\boldsymbol{V}_Q\| = 1$. That ends the first part of our proof.

ii) By definition, $\cos\theta = (\boldsymbol{V}_D \cdot \boldsymbol{V}_Q)/(\|\boldsymbol{V}_D\| \cdot \|\boldsymbol{V}_Q\|)$. Since the denominator is one (using the result proved above), we only need to be concerned about the upper part. Now, the upper part of the equation actually is,

$$\begin{aligned}
\boldsymbol{V}_D \cdot \boldsymbol{V}_Q &= s_1 d_{11} \cdot r_1 q_{11} + s_1 d_{12} \cdot r_1 q_{12} + \cdots + s_1 d_{1m} \cdot r_1 q_{1m} \\
&\quad + s_2 d_{21} \cdot r_2 q_{21} + s_2 d_{22} \cdot r_2 q_{22} + \cdots + s_2 d_{2m} \cdot r_2 q_{2m} \\
&\quad \vdots \\
&\quad + s_n d_{n1} \cdot r_n q_{n1} + s_n d_{n2} \cdot r_n q_{n2} + \cdots + s_n d_{nm} \cdot r_n q_{nm} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m} s_i d_{ij} r_i q_{ij} \\
&= \sum_{i=1}^{n} s_i r_i \sum_{j=1}^{m} d_{ij} q_{ij} \\
&= \sum_{i=1}^{n} s_i r_i (\boldsymbol{d}_{i*} \cdot \boldsymbol{q}_{i*}).
\end{aligned}$$

In other words, $\cos\theta = (\boldsymbol{V}_D \cdot \boldsymbol{V}_Q)/(\|\boldsymbol{V}_D\| \cdot \|\boldsymbol{V}_Q\|) = 1(\boldsymbol{V}_D \cdot \boldsymbol{V}_Q) = \sum_{i=1}^{n} s_i r_i (\boldsymbol{d}_{i*} \cdot \boldsymbol{q}_{i*})$. Proof ends.

Appendix B: The Manual for Encoding Prepositions as Thematic Role Triggers

B.1. General Rules

1. Only prepositions need to be encoded;
2. Emphasis should not be on the meaning of a preposition itself, but on the role it plays in signifying the relationship between the predicate and the object of the preposition;
3. If the object of a preposition happens to be a relative pronoun and is moved to the beginning of a clause but the preposition is left at the end, code it as ''functional'';
4. Code all particles as ''PARTICLE'';
5. Code the infinitive verb phrase marker ''to'' as ''infinitive.''

*B.2. List of Prepositions**

about	like
above	near
across	nearby
after	off
against	on
along	onto
alongside	out
among	outside
around	per
as	since
at	till
away	than
before	through
behind	throughout
below	to
beneath	toward
beside	towards
between	under
beyond	underneath
by	unlike
down	until
during	up
for	upon
from	via
in	with
inside	within
into	without

* The preposition ''of'' is not included since it often appears within a noun phrase but rarely functions as a direct trigger of case categories.

B.3. Definition and Illustration

ACCOMPANIMENT	Semantic unit or item that has the same relation to the predicate (the action or operation designated by the verb) as the main one has. e.g., I went there *with John.*
ACTION	Operation described by the predicate verb. Reserved for the main verb of a sentence. e.g., He *kicked* the ball really hard.
ALTERNATIVE	As it literally means. e.g., Drinking *instead of smoking.*
ATTRIBUTE	Quality of the entity that the prepositional phrase is meant to modify. e.g., The man *with a sense of humor.*
ASPECT	As it literally means. e.g., I'm not good *at associating with girls.*
BENEFICIARY	Entity or being for whose sake or benefit the event described by the sentence occurs. Often triggered by the preposition ''for.'' e.g., Please get a chair *for me.*
CAUSE	Factor or event that causally relates to the event described in the sentence. e.g., I do this *for all kinds of reasons.*
COMPARISON	As it literally means. e.g., This one differs *from that one.*
COMPONENT	As it literally means. e.g., The man *with a big nose.*
CONDITION	Event or contextual/environmental setting that is present for the event in question to happen. e.g., *With the lights on,* we now can see very clearly.
CONTENT	As it literally means. e.g., Here is an article *about abortion.*
CONSEQUENCE	Effect of the event referred to by the predicate. e.g., He opened the door *with the lock damaged.*
DESTINATION	Position or state where the motion ends. e.g., Get *to that point.*
DIRECTION	As it literally means. e.g., Lood *at him* fiercely.
EXTENT	As it literally means. e.g., It's true *to a large extent.*
EXPERIENCER	Entity or being whose emotional, psychological, or internal state is changed by the event described by the sentence. e.g., Your bright smile makes *me* happy.
PURPOSE	As it literally means. e.g., Work hard *for a better life.*
INSTRUMENT	Entity or item that is used as a tool in the event. e.g., You can write *with your finger.*
LOCATION	Place where the event in question happens or where the entities or beings involved in the event are. e.g., An explosion happened *in New York.*
MANNER	How the operation or action designated by the predicate happens. e.g., Eat *with grace,* and die *with dignity.*
MEANS	Way, process, or action through which the event in question happens. e.g., You can do it *by blowing this up.*
PATIENT	Entities or beings to or upon whom the event happens. e.g., We'll just have to fix *the brakes* again.
PROCESS	As it literally means. e.g., *In the process of doing this . . .*
QUANTITY	As it literally means. e.g., This thing costs three bucks.
RECIPIENT	As it literally means. e.g., Please return this book *to Tom.*

REFERENCE	As it literally means. e.g., Five miles away *from the town.*
SOURCE	Origin, starting point, or the place where an entity or being comes from. e.g., The data was generated *from the old sample of cockroaches.*
STATE	Current state, status, mode, or condition that an entity, event, or being is in. e.g., I want to choose the tree *in full blossom.*
TIME	Point or duration in the time dimension. e.g., Let's meet *at five o'clock.*
UNDEFINED	Whatever does not fit into any of the specific categories given above.

Appendix C: Proof of Ranking Effectiveness Ratio

C.1. Proof of Lower Bound

For a given query that has m relevant documents in the collection, because of the way ranking numbers are assigned to these m documents both in the relevance judgment base and on the system's output list, the base ranks of these m documents are actually a sequence of the natural numbers 1, 2, 3, . . . , m and the system ranks of these documents are the same numbers from 1 up to m, but in different order. For convenience of exposition, let d_1 be the document with base rank 1, d_2 with the base rank 2, $\cdots$ d_m with the base rank m. In general, let r_i be the base rank of the document d_i, and s_i be the system rank of document d_i, $i = 1, 2, \ldots, m$.

It is easy to see that the worst case of system ranking happens when a system rank sequence is completely the reverse of the base rank sequence, i.e., $s_1 = m$, $s_2 = m - 1$, $s_3 = m - 2$, . . . , $s_m = 1$. Now we proceed to calculate the ranking effectiveness ratio of this system rank sequence by first considering the case $m = 2k$ and then the case $m = 2k + 1$, $k = 1, 2, \ldots, n$.

C.1.1. The case m = 2k Since we have $s_1 = m$, $s_2 = m - 1$, $s_3 = m - 2$, . . . , $s_m = 1$, and $r_1 = 1$, $r_2 = 2$, . . . , $r_m = m$, the ranking effectiveness ratio of this system rank sequence can be calculated as follows.

$$\mu = \frac{2k\cdot 1 + (2k-1)\cdot 2 + (2k-2)\cdot 3 + \cdots + 2\cdot(2k-1) + 1\cdot 2k}{1^2 + 2^2 + \cdots + (2k-1)^2 + (2k)^2}$$

$$= \frac{2(2k\cdot 1 + (2k-1)\cdot 2 + (2k-2)\cdot 3 + \cdots + k\cdot(k+1))}{1^2 + 2^2 + \cdots + (2k-1)^2 + (2k)^2}$$

$$\mu = \frac{2(2k\cdot 1 + (2k-1)\cdot 2 + (2k-2)\cdot 3 + \cdots + (2k-k)\cdot(k+1))}{1^2 + 2^2 + \cdots + (2k-1)^2 + (2k)^2}$$

$$= \frac{2(2k\cdot 1 + 2k\cdot 2 - 1\cdot 2 + 2k\cdot 3 - 2\cdot 3 + \cdots + 2k\cdot(k+1) - k\cdot(k+1))}{1^2 + 2^2 + \cdots + (2k-1)^2 + (2k)^2}$$

$$= \frac{2[2k(1 + 2 + 3 + \cdots + (k+1)) - (1\cdot 2 + 2\cdot 3 + 3\cdot 4 + \cdots + k(k+1))]}{1^2 + 2^2 + \cdots + (2k-1)^2 + (2k)^2}$$

$$= \frac{4k(1 + 2 + 3 + \cdots + (k+1)) - 2(1\cdot 2 + 2\cdot 3 + 3\cdot 4 + \cdots + k(k+1))}{1^2 + 2^2 + \cdots + (2k-1)^2 + (2k)^2}$$

Note that,

1) $1 + 2 + 3 + \cdots + k + (k+1) = \frac{1}{2}(k+1)(k+2)$;

2) $1^2 + 2^2 + 3^2 + \cdots + (2k)^2 = \frac{1}{3}k(2k+1)(4k+1)$; and

3) $1\cdot 2 + 2\cdot 3 + 3\cdot 4 + \cdots + k\cdot(k+1) = \frac{1}{3}k(k+1)(k+2)$.

Replace the three series in the above expression with 1), 2), and 3), then

$$\mu = \frac{4k(1 + 2 + 3 + \cdots + (k+1)) - 2(1\cdot 2 + 2\cdot 3 + 3\cdot 4 + \cdots + k(k+1))}{1^2 + 2^2 + \cdots + (2k-1)^2 + (2k)^2}$$

$$= \frac{\frac{4k(k+1)(k+2)}{2} - \frac{2}{3}k(k+1)(k+2)}{\frac{1}{3}k(2k+1)(4k+1)}$$

$$= \frac{6k(k+1)(k+2) - 2k(k+1)(k+2)}{k(2k+1)(4k+1)}$$

$$= \frac{4\left(1 + \frac{1}{k}\right)\left(1 + \frac{2}{k}\right)}{\left(2 + \frac{1}{k}\right)\left(4 + \frac{1}{k}\right)}.$$

Therefore, the limitation of μ when $k \to +\infty$ is

$$\lim_{k\to+\infty} \mu = \lim_{k\to+\infty} \frac{4\left(1 + \frac{1}{k}\right)\left(1 + \frac{2}{k}\right)}{\left(2 + \frac{1}{k}\right)\left(4 + \frac{1}{k}\right)}$$

$$= \frac{4\left(1 + \lim_{k\to+\infty}\frac{1}{k}\right)\left(1 + \lim_{k\to+\infty}\frac{2}{k}\right)}{\left(2 + \lim_{k\to+\infty}\frac{1}{k}\right)\left(4 + \lim_{k\to+\infty}\frac{1}{k}\right)}$$

$$= \frac{4(1+0)(1+0)}{(2+0)(4+0)} = \frac{4}{8} = \frac{1}{2}$$

which means $\mu > \frac{1}{2}$ when $m = 2k$.

C.1.2. The case m = 2k + 1 Now consider the case $m = 2k + 1$. Similarly we have

$$\mu = \frac{(2k+1)\cdot 1 + ((2k+1)-1)\cdot 2 + ((2k+1)-2)\cdot 3 + \cdots + 2\cdot((2k+1)-1) + 1\cdot(2k+1)}{1^2 + 2^2 + \cdots + (2k)^2 + (2k+1)^2}$$

$$= \frac{2((2k+1)\cdot 1 + ((2k+1)-1)\cdot 2 + ((2k+1)-2)\cdot 3 + \cdots + (k+1)\cdot(k+1))}{1^2 + 2^2 + \cdots + (2k)^2 + (2k+1)^2}$$

$$= \frac{2((2k+1)\cdot 1 + ((2k+1)-1)\cdot 2 + ((2k+1)-2)\cdot 3 + \cdots + ((2k+1)-k)\cdot(k+1))}{1^2 + 2^2 + \cdots + (2k)^2 + (2k+1)^2}$$

$$= \frac{2\{(2k+1)\cdot 1 + (2k+1)\cdot 2 - 1\cdot 2 + (2k+1)\cdot 3 - 2\cdot 3 + \cdots + (2k+1)\cdot(k+1) - k\cdot(k+1)\}}{1^2 + 2^2 + \cdots + (2k)^2 + (2k+1)^2}$$

$$= \frac{2\{(2k+1)(1+2+3+\cdots+(k+1)) - (1\cdot 2 + 2\cdot 3 + 3\cdot 4 + \cdots + k(k+1))\}}{1^2 + 2^2 + \cdots + (2k)^2 + (2k+1)^2}$$

$$= \frac{2(2k+1)(1+2+3+\cdots+(k+1)) - 2(1\cdot 2 + 2\cdot 3 + 3\cdot 4 + \cdots + k(k+1))}{1^2 + 2^2 + \cdots + (2k)^2 + (2k+1)^2}$$

$$= \frac{2(2k+1)\dfrac{(k+1)(k+2)}{2} - \frac{2}{3}k(k+1)(k+2)}{\frac{1}{6}(2k+1)(2k+2)(4k+3)}$$

$$= \frac{3(2k+1)(k+1)(k+2) - 2k(k+1)(k+2)}{(k+1)(2k+1)(4k+3)}$$

$$= \frac{3(2k+1)(k+2) - 2k(k+2)}{(2k+1)(4k+3)}$$

$$= \frac{(k+2)(4k+3)}{(2k+1)(4k+3)} = \frac{k+2}{2k+1} = \frac{1+\frac{2}{k}}{2+\frac{1}{k}}$$

Therefore, the limitation of μ when $k \to +\infty$ is

$$\lim_{k\to+\infty} \mu = \lim_{k\to+\infty} \frac{1+\frac{2}{k}}{2+\frac{1}{k}} = \frac{1+\lim_{k\to+\infty}\frac{2}{k}}{2+\lim_{k\to+\infty}\frac{1}{k}} = \frac{1+0}{2+0} = \frac{1}{2}$$

which means for $m = 2k + 1$, $\mu > \frac{1}{2}$. Proof ends.

C.2. Proof of Upper Bound

For a given number n of relevant documents D_1, D_2, D_3, . . . , Dn, let r_i and s_i be the base rank and system rank for the document D_i, respectively, $i = 1, 2, 3, \ldots, n$. Consider the sum of squares of the differences between the system ranks and base ranks of the n documents relevant to a given query,

$$P = \sum_{i=1}^{n} (s_i - r_i)^2 = \sum_{i=1}^{n} (s_i^2 + r_i^2 - 2s_i r_i) = \sum_{i=1}^{n} s_i^2 + \sum_{i=1}^{n} r_i^2 - \sum_{i=1}^{n} 2s_i r_i .$$

Note that because both the base ranks and system ranks of these documents are the same set of numbers {1, 2, 3, . . . , n}, we have $\Sigma_{i=1}^{n} s_i^2 = \Sigma_{i=1}^{n} r_i^2$. Therefore,

$$P = \sum_{i=1}^{n} s_i^2 + \sum_{i=1}^{n} r_i^2 - 2\sum_{i=1}^{n} s_i r_i = 2\sum_{i=1}^{n} r_i^2 - 2\sum_{i=1}^{n} s_i r_i \geq 0.$$

Thus, $\Sigma_{i=1}^{n} r_i^2 \geq \Sigma_{i=1}^{n} s_i r_i$, which means $\mu = (\Sigma_{i=1}^{n} s_i r_i)/(\Sigma_{i=1}^{n} r_i^2) \leq 1$. Proof ends.

C.3. Proof of Monotonicity

Similarly as in section C.2, we have

$$P = \sum_{i=1}^{n} s_i^2 + \sum_{i=1}^{n} r_i^2 - 2\sum_{i=1}^{n} s_i r_i = 2\sum_{i=1}^{n} r_i^2 - 2\sum_{i=1}^{n} s_i r_i;$$

$$\sum_{i=1}^{n} r_i^2 - \sum_{i=1}^{n} s_i r_i = \tfrac{1}{2}P; \quad \sum_{i=1}^{n} s_i r_i = \sum_{i=1}^{n} r_i^2 - \tfrac{1}{2}P;$$

which means,

$$\mu = \frac{\Sigma_{i=1}^{n} s_i r_i}{\Sigma_{i=1}^{n} r_i^2} = 1 - \frac{P}{2\,\Sigma_{i=1}^{n} r_i^2} .$$

Since $\Sigma_{i=1}^{n} r_i^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{1}{6}n(n+1)(2n+1)$ is a constant for any given n, $\mu = f(P)$ is a strictly decreasing linear function of P. We know that P is the sum of squares of the differences between the system

ranks and base ranks, which means P is monotonic of the difference between the system rank and base rank of a relevant document. Therefore, $\mu = f(P)$ is a monotonic function of the differences between the system ranks and base ranks for the n documents relevant to the given query. Proof ends.

Appendix D: Sample Results of Heuristic Syntax Parsing

Text

[1]	Directions in library networking. [Title]
[2]	Bibliographic control before and after MARC is reviewed.
[3]	The capability of keying into online systems brought an interdependence among libraries, the service centers that mediate between them, and the large utilities that process and distribute data.
[4]	From this has developed the basic network structure among libraries in the United States.
[5]	The independent development of major networks has brought problems in standardization and coordination.
[6]	The authors point out that while technology has led toward centralization of automated library services, new developments are now pushing toward decentralization.
[7]	Coordination is a requirement to avoid fragmentation in this new development.

Syntactic Structure

[1]	PREP_PHRASE::	in library networking
	UNPARSED CLAUSE::	directions
[2]	CLAUSE::	
	VERB::	is reviewed
	VPC_DIR::	marc
	PREP_PHRASE::	before and after
	UNPARSED CLAUSE::	bibliographic control
[3]	MAIN_STRUCT::	
	SUBJECT::	the capability of keying
	VERB::	brought
	VPC_DIR::	an interdependence
	CLAUSE::	
	VERB::	distribute
	VPC_DIR::	data
	CLAUSE::	
	VERB::	process
	PREP_PHRASE::	between them
	PREP_PHRASE::	among libraries
	PREP_PHRASE::	into online systems
	ORPHANS::	the service centers that mediate and the large utilities
[4]	CLAUSE::	
	VERB::	has developed
	VPC_DIR::	the basic network structure
	PREP_PHRASE::	in the united states
	PREP_PHRASE::	among libraries
	PREP_PHRASE::	from this
[5]	MAIN_STRUCT::	
	SUBJECT::	the independent development of major networks
	VERB::	has brought
	VPC_DIR::	problems
	PREP_PHRASE::	in standardization and coordination
[6]	MAIN_STRUCT::	
	SUBJECT::	the authors
	VERB::	point
	CLAUSE::	
	SUBJECT::	while technology
	VERB::	has led
	VPC_DIR::	services
	PREP_PHRASE::	toward decentralization
	PREP_PHRASE::	toward centralization and automated library
	ORPHANS::	out new developments are now pushing
[7]	MAIN_STRUCT::	
	SUBJECT::	coordination
	VERB::	is
	VPC_DIR::	a requirement
	PREP_PHRASE::	in this new environment
	INFINIT_PHRASE::	to avoid fragmentation

References

Allen, J. (1988). *Natural language understanding*, Menlo Park, CA: Benjamin/Cummings.

Anderson, J. M. (1971). *The grammar of case: Towards a localistic theory.* London: Cambridge University Press.

Bailin, A., & Grafstain, A. (1991). The assignment of thematic roles in Ojibwa. *Linguistics, 29,* 397–422.

Blake, F. R. (1930). A semantic analysis of case. In J. T. Hatfield, W. Leopold, & A. J. F. Zieglschmid (Eds.), *Curme volume of linguistic studies (language monographs 7)* (pp. 34–49). Baltimore: Linguistic Society of America.

Bowerman, M. (1990). Mapping thematic roles onto syntactic functions: Are children helped by innate linking rules. *Linguistics, 28,* 1253–1289.

Brajnik, G., Guida, G., & Tasso, C. (1988). IR-NLI II: Applying man-machine interaction and AI concepts to information retrieval. *Proceedings of 11th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 387–339). New York: Association for Computing Machinery.

Carrier-Duncan, J. (1985). Linking of thematic roles in derivational word formation. *Linguistic Inquiry, 16,* 1–34.

Chierchia, G. (1988). Structured meanings, thematic roles, and control. In G. Chierchia, B. H. Partee, & R. Turner (Eds.), *Properties, types, and meaning: Vol. 2. Semantic issues* (pp. 131–166). Boston: Kluwer Academic Publishers.

Chomsky, N. (1981). Lectures on government and binding. Dordrecht, The Netherlands: Foris.

Cohen, P. R., & Kjeldsen, R. (1987). Information retrieval by constrained spreading activation in semantic networks. *Information Processing & Management, 23(2),* 255–268.

Cooper, W. S. (1991). Some inconsistencies and misnomers in probabilistic information retrieval. *Proceedings of the 14th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 57–61). New York: Association for Computing Machinery.

Cutting, D., Kupiec, J., Pedersen, J., & Sibun, P. (1992). A practical part-of-speech tagger. *Proceedings of the Third Conference on Applied Natural Language Processing* (pp. 43–57). San Francisco: Morgan Kaufmann.

Cutting, D. R., Pedersen, J., & Halvorsen, P. K. (1991). An object-oriented architecture for text retrieval. *Intelligent Text and Image Handling: Conference Proceedings of RIAO '91* (pp. 285–298). New York: Elsevier.

Dillon, M., & Gray, A. S. (1983). FASIT: A fully automatic, syntactically based indexing system. *Journal of the American Society for Information Science, 34,* 99–108.

Doszkocs, T. E. (1986). Natural language processing in information retrieval. *Journal of the American Society for Information Science, 37,* 161–186.

Dowty, D. (1989). On the semantic content of the notion ''thematic role.'' In G. Chierchia, B. H. Partee, & R. Turner (Eds.), *Properties, types, and meaning: Vol. 2. Semantic issues* (pp. 69–130). Boston: Kluwer Academic Publishers.

Dowty, D. (1991). Thematic proto-roles and argument selection. *Language, 9(1),* 547–619.

Evans, D. A., Gunther-Webster, K., Hart, M., Lefferts, R. G., & Monarch, I. A. (1991). Automatic indexing using selective natural language processing and first order thesauri. *Intelligent Text and Image Handling: Conference Proceedings of RIAO '91.* New York: Elsevier.

Fagan, J. (1987a). Automatic phrase indexing for document retrieval. *Proceedings of the 10th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 91–110). New York: Association for Computing Machinery.

Fagan, J. (1987b). *Experiments in automatic phrase indexing for document retrieval: A comparison of syntactic and non-syntactic methods.* (Tech. Rep. No. 87-868). Ithaca, NY: Cornell University, Department of Computer Science.

Fagan, J. (1989). The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. *Journal of the American Society for Information Science, 40,* 115–132.

Filmore, C. J. (1966). A proposal concerning English prepositions. *Monograph Series on Languages and Linguistics, 19,* 19–34.

Filmore, C. J. (1968). The case for case. In E. Bach & R. Harms (Eds.), *Universal in linguistic theory* (pp. 1–90). New York: Holt, Rinehart and Winston.

Filmore, C. J. (1969a). Toward a modern theory of case. In D. A. Reibel & S. A. Schane (Eds.), *Modern studies in English: Readings in transformational grammar.* Englewood Cliffs, NJ: Prentice-Hall.

Filmore, C. J. (1969b). Types of lexical information. In F. Kiefer (Ed.), *Studies in syntax and semantics.* Dordrecht: Reidel.

Filmore, C. J. (1970). Lexical entries for verbs. In M. Lester (Ed.), *Readings in applied transformational grammar.* New York: Holt, Rinehart and Winston.

Filmore, C. J. (1971). Some problems for case grammar. *Monograph Series on Languages and Linguistics, 24,* 35–56.

Filmore, C. J. (1982). Towards a descriptive framework for Deixis. In R. Jarvella & W. Klein (Eds.), *Speech, place, and action* (pp. 31–52). New York: Wiley.

Fuhr, N., & Buckley, C. (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems, 9(3),* 223–248.

Hahn, U., & Reimer, U. (1986). *Topic essentials.* (Tech. Rep. TOPIC-19/86). Konstanz, FRG: Universitaet Konstanz.

Jackendoff, R. (1972). *Semantic interpretation in generative grammar.* Cambridge, MA: MIT Press.

Jackendoff, R. (1990). *Semantic structure.* Cambridge, MA: MIT Press.

Lewis, D. D., & Croft, W. B. (1990). Term clustering of syntactic phrases. *Proceedings of the 13th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 385–404). New York: Association for Computing Machinery.

Lin, X. (1991). A self-organizing semantic map for information retrieval. *Proceedings of the 14th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 262–269). New York: Association for Computing Machinery.

Liu, G. Z. (1994). The semantic vector space model (SVSM): A text representation and searching technique. *Proceedings of the 27th Annual Hawaii International Conference on System Sciences, Vol. 4* (pp. 928–937). Los Alamitos, CA: IEEE Computer Society Press.

Lochbaum, K. E., & Streeter, L. A. (1989). Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval. *Information Processing and Management, 25(6),* 665–676.

Metzler, D. P. & Hass, S. W. (1989). The constituent object parser: Syntactic matching for information retrieval. *Proceedings of the 12th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 117–126). New York: Association for Computing Machinery.

Raghavan, V., & Wong, S. K. M. (1986). A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science, 37,* 279–287.

Rau, L. F. (1987). Knowledge organization and access in a conceptual information system. *Information Processing and Management, 23*(4), 269–283.

Rau, L. F., Jacobs, P. S., & Zernik, U. (1989). Information extraction and text summarization using linguistic knowledge acquisition. *Information Processing and Management, 25(4),* 419–428.

Rosen, C. (1984). The interface between semantic roles and initial grammatical relations. In Perlmutter, D., & Rosen, C. (Eds.), *Studies in relational grammar: Vol. 2* (pp. 38–77). Chicago: University of Chicago Press.

Sacks-Davis, R., Wallis, P., & Wilkinson, R. (1990). Using syntactic analysis in a document retrieval system that uses signature files. *Proceedings of the 13th ACM SIGIR Conference on Research and Devel-*

opment in Information Retrieval (pp. 179–192). New York: Association for Computing Machinery.

Salton, G. (1968). *Automatic information organization and retrieval.* New York: McGraw-Hill.

Salton, G. (Ed.). (1971). *The SMART retrieval system-experiments in automatic document processing.* Englewood Cliffs, NJ: Prentice-Hall.

Salton, G. (1989). *Automatic text processing: The transformation, analysis, and retrieval of information by computer.* Reading, MA: Addison-Wesley.

Salton, G. (1991). The SMART document retrieval projects. *Proceedings of the 14th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 357–378). New York: Association for Computing Machinery.

Salton, G., & Buckley, C. (1991). Global text matching for information retrieval. *Science, 253(5023),* 1012–1014.

Salton, G., & McGill, M. (1983). *Introduction to modern information retrieval.* New York: McGraw Hill.

Schwarz, C. (1990). Automated syntactic analysis of free text. *Journal of the American Society for Information Science, 41,* 408–417.

Sheridan, P., & Smeaton, A. F. (1992). The application of morpho-syntactic language processing to effective phrase matching. *Information Processing & Management, 28(3),* 349–369.

Smeaton, A. F. (1986). Incorporating syntactic information into a document retrieval strategy: An investigation. *Proceedings of the 9th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 103–113). New York: Association for Computing Machinery.

Smeaton, A. F. (1989). Information retrieval and natural language processing. In K. P. Jones (Ed.), *Prospects for intelligent retrieval: Informatics 10* (pp. 1–14). London: Aslib.

Smeaton, A. F., & Sheridan, P. (1990). *Using morpho-syntactic language analysis in phrase matching.* (Tech. Rep. SIMPR-DCU-1990-16.9e). Dublin: School of Computer Applications, Dublin City University.

Smeaton, A. F., & van Rijsbergen, C. J. (1988). Experiments on including syntactic processing of user queries into a document retrieval system. *Proceedings of the 11th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 31–51). New York: Association for Computing Machinery.

Smeaton, A. F., Voutilainen, A., & Sheridan, P. (1990). *The application of morpho-syntactic language processing to effective text retrieval.* (Tech. Rep. SIMPR-DCU-1990-16.5e). Dublin: School of Computer Applications, Dublin City University.

Sparck-Jones, K. (1971). *Automatic keyword classification for information retrieval.* London: Butterworths.

Sparck-Jones, K., & Kay, M. (1973). *Linguistics and information science.* New York: Academic Press.

Sparck-Jones, K., & Tait, J. I. (1984). Automatic search term variant generation. *Journal of Documentation, 40,* 50–66.

Sutcliffe, R. F. E. (1991). Distributed representations in a text based information retrieval system: A new way of using vector space model. *Proceeding of the 14th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 123–132). New York: Association for Computing Machinery.

Thiel, U., & Hammwohner, U. (1987). Informational zooming: An interaction model for the graphical access to text knowledge bases. *Proceedings of the 10th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 45–56). New York: Association for Computing Machinery.

Turtle, H., & Croft, W. B. (1991). Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems, 9(3),* 187–222.

Wendlandt, E., & Driscoll, J. R. (1991). Incorporating a semantic analysis into a document retrieval strategy. *Proceeding of the 14th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 270–279). New York: Association for Computing Machinery.

Wilkinson, R., & Hingston, P. (1991). Using the cosine measure in a neural network for documental retrieval. *Proceeding of the 14th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 202–210). New York: Association for Computing Machinery.

Wong, S. K. M., Ziarko, W., Raghavan, V. V., & Wong, P. C. N. (1987). On modeling of information retrieval concepts in vector spaces. *ACM Transactions on Database Systems, 12(2),* 299–321.

Wong, S. K. M., Ziarko, W., & Wong, P. C. N. (1985). Generalized vector space model in information retrieval. *Proceedings of the 8th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 18–25). New York: Association for Computing Machinery.

Wood, M., & Sommerville, I. (1988). An information retrieval system for software components. *ACM SIGIR Forum, 22(3/4),* 11–28.

Yao, Y. Y., & Wong, S. K. M. (1991). Preference structure, inference and set-oriented retrieval. *Proceedings of the 14th ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 211–217). New York: Association for Computing Machinery.

Zaring, L. (1991). On prepositions and case-marking in French. *Canadian Journal of Linguistics, 36(4),* 363.

Zernik, U., & Jacobs, P. (1990). Tagging for learning: Collecting thematic relations from corpus. H. Karlgren (Ed.), *Proceedings of COLING-90, Vol. 1* (pp. 34–39). Helsinki: Helsingiensis Universitas.